02

# Ctrl + Z en Git

## Transcripción

[00:00] Hola a todos y todas, bienvenidos y bienvenidas una vez más a este curso de Git. Ya podemos conectarnos con uno o más repositorios remotos, podemos compartir nuestro código con colegas de equipo, podemos organizar nuestro versionamiento en branches y líneas de desarrollos diferentes.

[00:15] Entonces ya hemos visto bastantes cosas. Ahora me di cuenta que acá en el repositorio de Ana no configuramos que el nombre de quien está haciendo commits sea Ana. Entonces para mantener un histórico correcto vamos a hacer eso. Vamos a hacer un git config --local user.name "Ana", enter. Perfecto. Ya configuramos.

[00:41] Bueno, continuando. Es muy común que cuando empezamos a desarrollar hagamos algunos tests y después necesitemos deshacer esas modificaciones. ¿Cómo podemos deshacer las modificaciones utilizando Git? ¿Será que Git tiene alguna especie de "Ctrl + Z"? Primero vamos a trabajar en el proyecto de Bruno y acá en Ansible quiero modificar por ejemplo el nombre y quiero colocar Ansible: Infraestructura como código, por ejemplo. Damos un "Ctrl + S".

[01:17] Acá es solamente un archivo, entonces si yo quisiera deshacer lo que hice, simplemente haciendo un "Ctrl + Z" puedo ir deshaciendo todo eso. Pero imaginen que ustedes están en un proyecto mucho más grande, donde tienen muchas modificaciones para probar y vemos que no es lo que nosotros queremos. ¿Necesitamos ir por cada uno de esos archivos haciendo "Ctrl + Z" en todos?

[01:41] Imaginen por ejemplo que hice modificaciones ayer. Pero solo las puedo probar hoy y no me gustaron. ¿Git me ayudará a deshacer esas modificaciones? Primero voy a hacer un "Ctrl + Shift + Z" para ir hacia adelante en el código y vamos a irnos a Git, a nuestro Git de Bruno, a la terminal de Bruno, y vamos a hacer un git status, enter, y vean lo que nos está diciendo: hay cambios que no están agregados para hacer un commit. [02:17] Entonces si queremos agregarlos, hacemos un git add pero si por ejemplo nosotros queremos descartar los cambios en nuestra carpeta, podemos utilizar git restore y el nombre del archivo. Antiguamente, esto se utilizaba con un checkout. Nosotros podemos hacer un git checkout -- y después el nombre del archivo y funcionaba igual que este restore. En esta versión que estoy utilizando se ha modificado por este restore.

[02:46] Entonces podemos hacer un git restore y el nombre del archivo en este caso es index.html. Damos un enter, no nos dice nada, pero si nos vamos nosotros al proyecto, vean que no está más ese cambio que hicimos de Ansible. Pero imaginemos ahora que vamos a hacer el cambio ese de Ansible: Infraestructura como código, por ejemplo. Damos un "Ctrl + S" e imaginen que acá estando en Bruno, voy a hacer un clear. Y nosotros hacemos un git add index.html.

[03:23] Vamos a suponer que esa es la situación. Hagamos un git status. Por ejemplo imagínense que nosotros nos dimos cuenta que antes de hacer un commit deberíamos testear, por ejemplo. Entonces vemos que acá nos está diciendo que si nosotros ya tenemos cambios para hacer un commit, ya hicimos el git add y queremos volver para atrás, lo que podemos utilizar es este comando git restore --staged, espacio y el nombre del archivo.

[03:57] Entonces vamos a hacer eso: git restore --staged, y en este caso index.html, vamos a dar un enter y ahora vamos a dar un git status. Vean que lo que ha hecho es sacar el add que le habíamos dejado y las modificaciones siguen estando. Si yo me voy al archivo, seguimos teniendo esto, es como si nunca hubiéramos hecho el git add. Y por ejemplo si necesitamos realmente

sacar estas modificaciones, es simplemente utilizar el mismo comando anterior.

[04:33] Que el comando anterior es git restore index.html. Damos enter y vean que acá en el archivo ahora ya no tenemos las modificaciones. Pero ahora, imaginen el peor de los casos, vamos a hacer la modificación de Ansible: Infraestructura como código. Vamos a dar un "Ctrl + S", vamos a estar acá en Bruno, vamos a hacer un clear de la pantalla, git add index.html. Ahora vamos a hacer un commit, git commit -m, y acá vamos a poner: "Cambiando el nombre del curso de Ansible". Enter.

[05:27] Dimos el commit. Pero después testeando vi que puse un bug ahí, vi que el código no debería haber sido commiteado. ¿Cómo puedo dar un "Ctrl + Z" a un commit que ya hice? Bueno, vamos a ir a nuestro log haciendo un git log, damos enter, y vean que nosotros tenemos ese ID que es un hash, que nos está indicando cuál es el commit que hemos hecho.

[05:51] Entonces voy a copiar ese commit, "clic derecho + copiar", vamos a dar la letra Q para salir del log y entonces lo que vamos a hacer es colocar git revert, espacio, y vamos a hacer un clic derecho, pegar, enter y vean que lo que va a hacer en realidad es crear un nuevo commit deshaciendo el commit anterior. Entonces por ejemplo tenemos acá para cambiar el mensaje si queremos sin problema, podemos poner lo que queramos.

[06:24] Si estamos de acuerdo con el mensaje del nuevo commit, vamos a apretar :x, damos enter, y ahora vamos a ir a nuestro log. Vamos a hacer un git log, enter y vean lo siguiente: tenemos nuestro commit, cambiando el curso, y tenemos el commit que revierte ese commit.

[06:45] Entonces si vamos a nuestro archivo, vemos que ya no están más los cambios que habíamos hecho. Y con esto vemos que en cada uno de esos diferentes pasos, tanto si no hemos hecho add, si hicimos el add, si hicimos el

commit, en cada uno de esos casos podemos ir volviendo para atrás en nuestros pasos e ir revirtiendo cada uno de ellos.

[07.10] Pero imagina que hicimos algunos tests, escribimos un código, pero ese código aun no está listo para commitear, que está en una etapa que aún no funciona, pero no quiero deshacerlo. Llegó una tarea nueva para hacer por ejemplo. ¿Y será que puedo guardar temporalmente ese código para después? Bueno, vamos a discutir eso en el próximo video.